Learning Semantic Part-Based Models from Google Images

Learning Semantic Part-Based Models from Google Images

Davide Modolo and Vittorio Ferrari

Abstract—We propose a technique to train semantic part-based models of object classes from Google Images. Our models encompass the appearance of parts and their spatial arrangement on the object, specific to each viewpoint. We learn these rich models by collecting training instances for both parts and objects, and automatically connecting the two levels. Our framework works incrementally, by learning from easy examples first, and then gradually adapting to harder ones. A key benefit of this approach is that it requires no manual part location annotations. We evaluate our models on the challenging PASCAL-Part dataset [1] and show how their performance increases at every step of the learning, with the final models more than doubling the performance of directly training from images retrieved by querying for part names (from 12.9 to 27.2 AP). Moreover, we show that our part models can help object detection performance by enriching the R-CNN detector with parts.

Index Terms—Part Detection, Web Learning, Curriculum Learning

1 Introduction

Part-based models have gained significant attention in the last few years. The key advantages of exploiting part representations is that parts have lower intra-class variability than whole objects, they deal better with pose variation and their configuration provides useful information about the aspect of the object. Parts localization has therefore been addressed in the context of several vision tasks, such as object recognition [1], [2], [3], object segmentation [4], [5], fine-grained classification [6], [7], [8], human pose-estimation [9], [10], [11], attribute prediction [12], [13], action classification [15] and scene classification [16], achieving state-of-the-art results in many of them.

Part-based methods can be grouped into two sets. The first set of works define an object part as any patch that is discriminative for the object class [2], [3], [4], [5], [16]. These works typically discover parts in the training images automatically, without human supervision. However, their resulting parts do not have a meaning for humans (e.g. a patch straddling between the wheel and the chassis of a car [3]). The second set of works define parts semantically (e.g. 'wheel') [1], [5], [6], [7], [9], [10], [11], [13]. These are more interpretable for a human and are necessary to obtain fine descriptions of objects and their interactions. For example, "the headlights of the bus are turned on" and "the cat is touching the TV with its tail". Moreover, part localization is necessary for a robot to correctly grasp an object (e.g. grasp a mug by the handle). However, existing works on semantic part detection require part location annotations in the training images, which are very expensive to obtain.

In this paper we try to get the best of both worlds by proposing a novel method to train semantic part models of object classes without manual location annotations. We train these models on images automatically collected from Google Images. We represent an object class as a mixture over multiple viewpoints. We learn a collection of semantic part appearance models, and models of their spatial arrangement on the object, specific to each viewpoint.

• *D. Modolo and V. Ferrari are with the IPAB institute at the University of Edinburgh. E-mail for correspondence: davide.modolo@gmail.com*

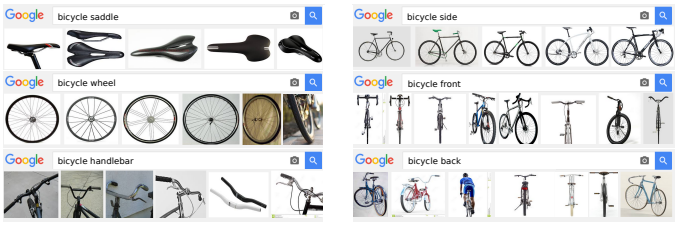


Fig. 1: *Images returned from Google Images. On the left examples of queries for object parts, while on the right queries for an object under different viewpoints. Note how the instances are correct, clean and they mostly appear under a uniform background.*

Moreover, we also train models capable of predicting the viewpoint of the object, which we then use to select an appropriate location model to guide part localization on novel instances. Learning from the web has been addressed before [17], [18], [19], [20], [21], [22], [23], [24], [25], but mostly at the level of object classes. Instead, here we learn complex semantic part-based models from the web.

We learn these rich models fully automatically, entirely from Google Images, by collecting training instances for both *parts* and *objects* (fig. 1), and automatically connecting the two levels. Our technique incrementally learns from easy examples first, and then gradually adapts to harder examples. This adaptation is done within Google Images, where part images offer easy examples (fig. 1 left), and then harder examples are mined from object instances (fig. 1 right). The move from part images to object images also enables us to learn the spatial arrangement of parts on the object (location models). In a final step, we further adapt our models on an external, non-Google image domain to adapt to even harder examples, e.g. on PASCAL VOC [26].

We demonstrate the effectiveness of our incremental learning algorithm on the PASCAL-Part dataset [1]. Interestingly, the performance of our part models increases at every step of the learning, with the final models more than doubling the initial performance (from 12.9 to 27.2 AP). Moreover, we compare to two other webly-supervised works (LEVAN [23] and NEIL [22]) and show that our part models perform better. Finally, we also show that our part models can help object detection performance by enriching

the R-CNN detector [27] with parts.

2 Related work

Many works learn non-semantic part models, where the parts are arbitrary patches that are discriminative for an object class [2], [3], [4], [5], [16]. Our work is more related to semantic part-based models, and to techniques for learning object classes from image search engines.

Semantic part-based models. There is a considerable amount of work on using semantic parts to help recognition tasks. The largest part of this has focused on the fine-grained recognition problem in several animal domains, such as birds [8], [9], [13], [28] and pets [6], [7]. In these works, an object is treated as a collection of parts that models its shape and appearance. Semantic parts help capturing subtle object appearance differences that could not be captured by a monolithic object model. These differences are crucial to discriminate between animal breeds. Other applications where semantic part-based models have been used are object detection [1], articulated human and animal pose estimation [9], [10], [11] and attribute prediction [13], [15]. In object detection, parts help dealing with deformed, occluded and low resolution objects. In articulated pose estimation, parts help identifying objects in special configuration (e.g. jumping and sitting) as opposed to canonical ones. Finally, in attribute prediction, attributes are predicted best by the part containing direct evidence about them.

All the above mentioned methods require accurate part location annotations for training (either in terms of keypoints [6], [9] or bounding boxes [1], [7], [8], [10], [11], [13], [15], [28]). Our framework instead does not require annotations of part positions nor extent, and automatically learns from Google Images instead.

Learning from image search engines. Several works have tried to learn visual models from training samples collected automatically from image search engines [17], [18], [19], [20], [21], [22], [23], [24], [25]. Most of them tackle image classification [17], [18], [19], [20], [21] and develop algorithms to find good training samples and learn iteratively.

Some works try to learn object class detectors from the web [22], [23], [24]. Chen *et al.* [24] considers only objects and not parts. LEVAN [23] leverage Google Books Ngrams to discover all appearance variations of an object class, then trains an object detector with a separate component per variation. While some of the components happen to represent parts (e.g. 'horse head'), these are treated just like other independent components (at the same level as 'jumping horse' and 'racing horse'). NEIL [22] mines web images to discover common sense relationships between object classes (e.g. 'car is found in raceway'), including also some part-of relations (e.g. 'wheel is part of car'). Importantly, both LEVAN and NEIL learn simple object class detectors, consisting of 'root filters' only. Instead we learn more complex, structured models of object classes, which include semantic part appearance models and their spatial arrangements within the object, conditioned on object viewpoint. Note how [22], [23], [24] do not report quantitative localization results for part detection. Finally, we believe our work is complementary to [22], [23], [24]. The frameworks of [22], [23] could provide a list of which parts belong to which object class, which could be passed on to our technique to learn more complex models. Moreover, our part models could be used in combination with the strong R-CNN root filters learned from the web by [24] (analog to sec. 5.4).

The concurrent work [25] is the only other one to learn semantic part models from the web. It learns the structure of objects in an embedding space where geometric relationships are implicitly conveyed by non-semantic mid-level parts. Instead, we learn explicit relationships on the semantic parts themselves, based on object and parts instances automatically mined from web images.

3 Overview of our approach

We present here an overview of our framework for automatically learning compositional semantic part models. We learn these models for each object class separately. For each class, we use Google Images to collect images of the object under several pre-defined viewpoints, and images of its parts. We use the part samples to train initial part appearance models, which are later used to learn the connection between the parts and the whole object. Learning this association is the key to our compositional part models.

For each class, we learn part appearance models $\mathcal{A}$, part location models $\mathcal{L}$, and object viewpoint classifiers $\mathcal{V}$. Our framework operates in four stages: $\mathcal{T}_0 - \mathcal{T}_3$ (fig. 2). In the first stage $\mathcal{T}_0$, we collect training samples from Google Images (objects and parts, fig. 1). Then, we iteratively learn the components of our part models, each time learning from harder examples: ($\mathcal{T}_1$) images containing only parts from Google, ($\mathcal{T}_2$) part examples mined from object images from Google, and ($\mathcal{T}_3$) part examples mined from object images from the PASCAL VOC 2010 dataset [26]. Every stage is fully automatic and does not require human intervention.

For each object part, we learn one appearance model and V location models, one for each viewpoint in our predefined set. A single location model is not sufficient to capture the position of a part with respect to the object, as this is strongly affected by viewpoint changes. For example, the front view of a bicycle has one wheel on the bottom-center of the bicycle, while a bicycle from the side has two wheels on the bottom left and right (fig. 1, right).

For simplicity, in the rest of the paper we use superscripts to indicate the stage a model component is trained at. For example, our part appearance model $\mathcal{A}^2$ is trained at stage $\mathcal{T}^2$, while $\mathcal{A}^3$ at stage $\mathcal{T}^3$.

$\mathcal{T}^0$: Collecting data. Our framework queries Google Images for images of an object under canonical viewpoints and images of each of its parts (sec. 4.1). These images are biased towards simple representations, in a uniform background and they reliably contain the wanted object (or part). However, one image may contain multiple instances or objects not appearing nicely in the centre (fig 3). It would be better if each object/part instance would be enclosed in a tight bounding-box. Bounding-boxes around parts help learning accurate appearance models as they exclude background pixels, and around objects they help learning accurate part location models as they provide a stable coordinate frame common to all instances. We therefore devise a simple, yet effective algorithm to fit a tight bounding-box around each part/object instance

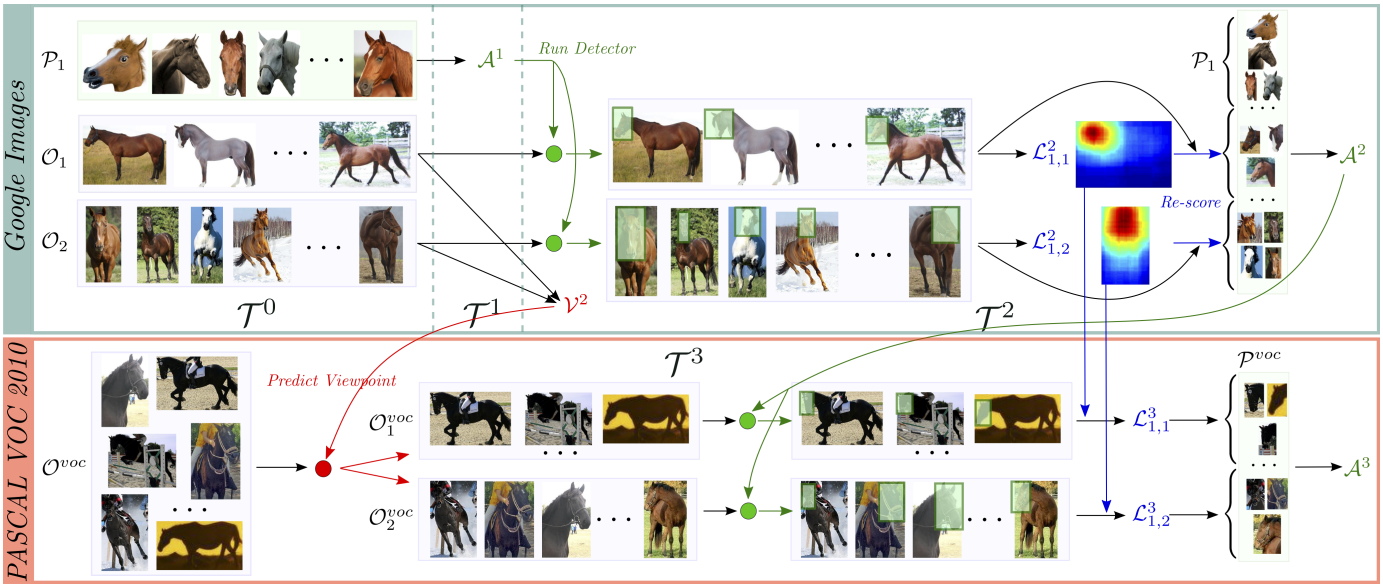


Fig. 2: *Schema of our framework for object class 'horse', simplified to just one part 'head' and two viewpoints 'left' and 'front'. At $\mathcal{T}^0$ the framework downloads horse instances ($\mathcal{O}_1$ and $\mathcal{O}_2$, for the two viewpoints) and head instances ($\mathcal{P}_1$) from Google Images. At $\mathcal{T}^1$ it learns a first head appearance model $\mathcal{A}^1$. At $\mathcal{T}^2$ it then hunts for new head instances from the horse images in $\mathcal{O}_1$ and $\mathcal{O}_2$ to first train two head location models $\mathcal{L}^2_{1,1}$ and $\mathcal{L}^2_{1,2}$, one for each horse viewpoint, and later to re-train a more accurate head appearance model $\mathcal{A}^2$. Finally, it also learns a viewpoint classifier $\mathcal{V}^2$. At $\mathcal{T}^3$ it then predicts the viewpoint of objects in $\mathcal{O}_{voc}$ using $\mathcal{V}^2$ and hunts for more part instances from them. These are then used to train our final part appearance model $\mathcal{A}^3$ and part location models $\mathcal{L}^3_{1,1}$ and $\mathcal{L}^3_{1,2}$. Note how at time $\mathcal{T}^1$ the framework has only seen part instances and has no information to learn neither $\mathcal{L}^1$ nor $\mathcal{V}^1$.*

(sec. 4.1). Finally, we consider each bounding-box as a separate image, obtaining our initial training set. We denote with $\mathcal{O}_j \in \mathcal{O}$ the set of images of the object under viewpoint v_j and with $\mathcal{P}_i \in \mathcal{P}$ the set of images of part p_i.

$\mathcal{T}^1$: Learning from Google's easy examples. For each part p_i, our framework learns an appearance model $\mathcal{A}^1_i$ on the part images $\mathcal{P}_i$ (sec. 4.2). These are the easiest samples, as in these images the part often appears isolated from the object and against a clean background (fig. 1 left).

$\mathcal{T}^2$: Learning from Google's harder examples. In this stage our framework moves on to object images $\mathcal{O}$. It learns part location models $\mathcal{L}^2$ and updates all part appearance models by using additional samples from $\mathcal{O}$ (sec. 4.2). Moreover, it trains an object viewpoint classifier $\mathcal{V}^2$ on $\mathcal{O}$ (sec. 4.4).

For each viewpoint v_j and part p_i, it learns $\mathcal{L}^2_{i,j}$. The key idea is to run $\mathcal{A}^1_i$ on the object images $\mathcal{O}_j$. The top-scored part detections are likely to be correct and, importantly, they are now *localized within an object image*. Therefore, they provide valuable training samples for the location of the part within the object (sec. 4.3). The intuition here is that objects captured under the same viewpoint have parts in similar spatial arrangements. For example, all horses from the side have the head on the left side of the image, mostly on top (fig. 2). Since all objects in $\mathcal{O}_j$ are in the same viewpoint v_j, correct detections of a part will be consistently found at similar locations across different object instances.

Subsequently, the framework mines part samples automatically from each $\mathcal{O}_j \in \mathcal{O}$ using $\mathcal{A}^1_i$ and the corresponding $\mathcal{L}^2_{i,j}$ (sec. 4.5). The process looks for detections that have a high score according to $\mathcal{A}^1_i$ and are at the right location according to $\mathcal{L}^2_{i,j}$. By combining these two source of information, we consistently discover correct part samples. Finally, the framework uses these samples to update part appearance models to $\mathcal{A}^2_i$. Note how these new samples are more difficult than the ones in $\mathcal{T}^1$, since come from images showing whole objects and against natural backgrounds. Lastly, the framework trains an object viewpoint classifier $\mathcal{V}^2$ on $\mathcal{O}$, by using each set of images $\mathcal{O}_j \in \mathcal{O}$ as training set for viewpoint v_j (sec. 4.4). Finally, note how at this stage the framework has trained a complete, rich model (part appearance $\mathcal{A}^2$, part location $\mathcal{L}^2$, object viewpoint $\mathcal{V}^2$) entirely and automatically from Google Images (fig. 2, top).

$\mathcal{T}^3$: Learning from PASCAL VOC. In this final stage the framework refines all $\mathcal{A}^2$ and $\mathcal{L}^2$ using even more difficult training samples from another domain (sec. 4.2, 4.3). These samples are mined automatically from the PASCAL VOC dataset, which contains photographs depicting challenging objects in natural scenes, often occluded or truncated (fig. 2, bottom). These are much harder than the ones in stage $\mathcal{T}^2$, where each image had a single whole object. The framework mines positives as in step $\mathcal{T}^2$, but using the updated $\mathcal{A}^2$ instead of the initial $\mathcal{A}^1$ (sec. 4.5). Similarly to other works [7], [28], we only search for parts inside the ground-truth bounding-boxes of the object class (which are provided with PASCAL VOC). We call this set $\mathcal{O}^{voc}$. Furthermore, we call the set of mined positives $\mathcal{P}^{voc}$. In order to use our viewpoint-specific location models, we need to determine the viewpoint of the images in $\mathcal{O}^{voc}$. We automatically predict v_j for each image in $\mathcal{O}^{voc}$ using the object viewpoint classifier $\mathcal{V}^2$. Finally, after mining new positives, the framework finally trains final location models $\mathcal{L}^3_{i,j}$ and part appearance models $\mathcal{A}^3_i$ (sec. 4.5).

4 The Components of Our Approach

We detail below the components of our approach. In sec. 4.1 we describe our data collection mechanism. In sec. 4.2, 4.3 and 4.4 we describe how to train $\mathcal{A}$, $\mathcal{L}$ and $\mathcal{V}$, respectively. In sec. 4.5 we then present our procedure to automatically mine new part instances from objects.

4.1 Data collection and preprocessing

This section describes how we download part and object images from Google and how we fit a tight bounding-box around each part/object instance in them.

Querying Google Images. We collect images of an object under multiple viewpoints and of its parts (fig. 1). We keep the top 100 retrieved images for each object viewpoint and the top 25 for each object part. We observed these numbers to produce good, clean images. Collecting more than 25 part images sometimes delivers spurious images without the part, which would introduce noise in the learning process.

For each object class, we use the names of its parts as listed in the PASCAL-Part Dataset [1] and the viewpoint names specified by PASCAL VOC 2010 [26] (*front*, *back*, *left*, *right*). As *left* and *right* is not a level of granularity satisfied by Google Images yet, we query for a generic *side* viewpoint (fig. 1 right-top) and then automatically split the retrieved images into left and right subsets. In order to do this, we first augment the image set by mirror flipping all images horizontally, and then we cluster them by minimizing the intra-cluster HOG compactness, similarly to [3].

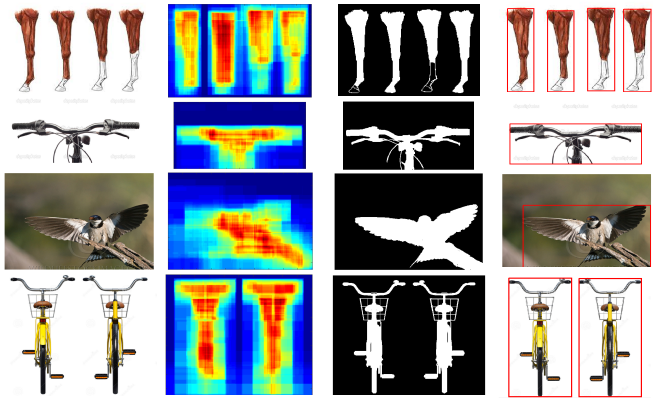
Fitting bounding-boxes. As mentioned in sec. 3, we want to fit a tight bounding-box around each object/part instance. These bounding-boxes help learning accurate appearance and location models for the parts. Fortunately, Google Images results are biased towards whole objects in a uniform background (fig. 3a) and unoccluded. These are easy to localize. We formulate this task as a pixel labelling problem, where each pixel ϕ_i can take a label $l_i \in \{0, 1\}$ (background or foreground). We aim at finding the best labelling $\psi^* = \text{argmin}_L E(\psi)$. Similar to other segmentation works [29], [30], [31], we define an energy function:

$$E(\psi) = \sum_i M_i(l_i) + \sum_i G_i(l_i) + \alpha \sum_{i,j} V(l_i, l_j), \qquad (1)$$

where, the pairwise potential V encourages smoothness by penalising neighbouring pixels taking different labels and the unary potential G_i evaluates how likely a pixel i is to take label l_i according to an appearance model which consists of two GMMs [29] (foreground and background). Inspired by [30], we produce an initial rough estimate M of which pixels lie on the object, and use it both to estimate the appearance models G, and as a unary potential of its own. We do this in an unsupervised manner, based purely on the spatial distribution of object proposals [32] in the image (fig. 3b). We define the likelihood $M_i(1)$ of a pixel i to be foreground as the number of proposals that contain it, divided by the total number of proposals in the image ($M_i(0) = 1 - M_i(1)$). The idea is that if a pixel is contained in many proposals, then it is likely to belong to the object.

As in GrabCut [29], we iteratively alternate between minimizing the energy (eq. 1) to obtain a segmentation, and updating the appearance models based on this segmentation. After a few iterations this process converges and we fit a tight bounding-box around each connected component in the final segmentation (fig. 3d). We apply this procedure to all images collected from Google, obtaining the initial training set of object images $\mathcal{O}$ and part images $\mathcal{P}$ (treating each bounding-box as a separate image).

Part proposals. We generate class-independent part proposals inside each image in $\mathcal{O}$ using [32]. As observed by Zhang *et al.* [8], these proposals achieve low recall on small semantic parts. In order to overcome this difficulty, we changed the standard settings of [32] to return smaller proposals and increase part recall. This results in about 2000 proposals per object image in $\mathcal{O}$, likely to cover all parts. In the rest of the paper we use $\mathcal{W}$ to refer to the set all part proposals over all object images.



(a) Input image (b) M (c) $E(\psi)$ (d) BB output

Fig. 3: *Examples of the steps of our procedure to fit bounding-boxes to object/part instances in web images (sec. 4.1). (a) is the input image; (b) is the initial rough foreground estimate M; (c) is the output of the segmentation process; and (d) are the bounding-boxes fit to connected components in the segmentation. Note how the two images 'horse leg' and 'bicycle front' have multiple part/object instances and our method is able to fit a separate bounding-box around each of them.*

4.2 Training part appearance models $\mathcal{A}$

Each stage of our learning framework updates the part appearance models of the object class. We describe here how these models are trained at each stage.

Stage $\mathcal{T}^1$. We train $\mathcal{A}^1$ on the image set $\mathcal{P}$, containing simple part images. As appearance model we use a convolutional neural network (CNN) and train it to distinguish between the P parts. More specifically, we start from AlexNet pre-trained on the ImageNet 2012 classification challenge [33] and replace its original 1000-way `fc8` classification layer with a P-way `fc8` layer. We then finetune the whole network for part classification on the images in $\mathcal{P}$. Note how $\mathcal{P}$ only contains 25 samples per part. In order to avoid overfit we use a learning rate of 10^{-4} and apply early stopping (1000 iterations, 5 epochs). Higher learning rates cause the parameters to vary abruptly over iterations, whereas 10^{-4} results in a smooth learning curve. At test time we use the softmax at layer `fc8` to predict how likely a proposal is to contain each of the parts. At all times we use the publicly available CNN implementation [34].

Stage $\mathcal{T}^2$. At this stage we learn $\mathcal{A}^2$ on a larger training set of examples from both part images and part samples automatically mined from object images using the appearance model from stage $\mathcal{T}^1$ and the location model from stage $\mathcal{T}^2$ (sec. 4.5, fig. 2). As appearance model we train a similar CNN to the one of stage $\mathcal{T}^1$, but with a difference: we use a richer $(P+1)$-way `fc8` layer, where the additional output is used to classify background patches. Note how by mining for positive part instances in object images (sec. 4.5) we indirectly discover negative proposals (those with intersection-over-union (IoU) [26] ≤ 0.3 with mined positives).

Fig. 4: *Examples of our location models $\mathcal{L}$. We show a canonical image of each object captured under one of our viewpoints and the location models of their parts. These models nicely capture the average position of each part within the object in that viewpoint. Note how these are automatically learnt from Google Images. For visualization, we show a 2D projection of the location models, which however live in a 4D space defined not only by the (x, y) position of a proposal, but also by its scale and aspect ratio.*

Stage $\mathcal{T}^3$. In the last stage we train $\mathcal{A}^3$ on the harder image set $\mathcal{P}^{voc}$, using as training samples parts automatically mined from $\mathcal{O}^{voc}$ using the part detector from stage $\mathcal{T}^2$ (sec. 4.5). As appearance model we train a CNN as in $\mathcal{T}^2$.

4.3 Learning part location models $\mathcal{L}$

The appearance model $\mathcal{A}$ scores part proposals in an image based on their appearance only. We build location models to capture complementary knowledge about likely positions and scales of the object parts within the the coordinate frame of the object. In stage $\mathcal{T}^2$ we learn the location models purely from Google Images and in stage $\mathcal{T}^3$ we adapt them to a different domain. In this subsection we use $\mathcal{W}_j$ to refer to the set of all part proposals in object images $\mathcal{O}_j$.

Part training samples. For each viewpoint v_j and part p_i, we learn a separate location model $\mathcal{L}_{i,j}$ from a set of training proposals $\mathcal{W}_{j,i} \in \mathcal{W}_j$ likely to contain part p_i. We describe here how we acquire these part samples $\mathcal{W}_{i,j}$ at stage $\mathcal{T}^2$, i.e. from object images $\mathcal{O}_j$. The key idea is to run the part detector $\mathcal{A}_i^1$ on these images and retain the top-scored part detections. As these detections are localized within an object image, they provide examples of the location of the part within the object. More precisely, for each image we score all part proposals with the appearance model $\mathcal{A}_i^1$, perform non-maximum suppression, and pick up to 3 detections per image (the top scored ones, if they score above a minimum confidence threshold). These detections form $\mathcal{W}_{i,j}$. This way of picking detections strikes a good trade-off between keeping all correct locations, but without including too many false-positives. At $\mathcal{T}^3$, we enrich the sample set $\mathcal{W}_{j,i}$ with the top detections produced by running the appearance model $\mathcal{A}_i^3$ on $\mathcal{O}^{voc}$. More specifically, each of these detections gets assigned to the $\mathcal{W}_{j,i}$ of viewpoint predicted by $\mathcal{V}^2$. These new samples are used to train the refined location models $\mathcal{L}_{i,j}^2$.

Training a location model. The location model $\mathcal{L}_{i,j}$ scores on an input part proposal w' by the density of the training set $\mathcal{W}_{j,i}$ at w'

$$\mathcal{L}_{i,j}(w') = \frac{1}{|\mathcal{W}_{j,i}| \cdot h} \sum_{w \in \mathcal{W}_{j,i}} K\left(\frac{D(w', w)}{h}\right) \quad (2)$$

where $D(w', w)$ is distance between two proposals: $D(w', w) = 1 - \frac{w' \cap w}{w' \cup w}$ and $K(u)$ is the uniform density function $K(u) = \frac{1}{2}\mathbb{1}(|u| \leq 1)$.

In this formulation $\mathcal{L}_{i,j}(w')$ has an intuitive interpretation as the percentage of proposals in $\mathcal{W}_{j,i}$ which are close to w' ($IoU < h$). If many training proposals are near w', then $\mathcal{L}_{i,j}(w')$ is large, indicating that w' is likely to contain the part. Conversely, if only a few proposal are near w', then $\mathcal{L}_{i,j}(w')$ is small, indicating it is more likely to cover a background patch. The bandwidth h controls the degree of smoothing and in our experiments we set it to $h = 0.5$.

Note how $D(w', w)$ compares part proposals across different images. For this to be meaningful, D operates in a coordinate frame common to all images in $\mathcal{O}_j$ (by normalizing it by the average width and height of all images). This normalization is specific to a viewpoint v_j, so it preserves its aspect-ratio.

Model behaviour. Fig. 4 shows examples of some location models learned at stage $\mathcal{T}^2$. Thanks to the way we build them, our location models are robust to errors in the training set: correct training samples tend to cluster around the right locations of a part, whereas incorrect ones tend to scatter across the whole object. This results in strong peaks at the correct locations in the model, with only lower values everywhere else (e.g. headlight for Bus Front). Moreover, note how our location model is suitable for a variety of cases. Unique parts of rigid objects form unimodal distributions (Bicycle saddle, Car license plate), while Bicycle wheel and Horse leg form bimodal ones. Even in the hard case of highly movable parts of deformable object classes (e.g. Cat tail), the model learns that they can appear over broader regions and spreads the density accordingly.

4.4 Training the viewpoint classifier $\mathcal{V}^2$

During stage $\mathcal{T}^2$ we train classifiers $\mathcal{V}^2$ on $\mathcal{O}$ for predicting the viewpoint of the object in an image. We train a CNN to distinguish between the four viewpoints (*front, back, left, right*) for which we collected object images from Google in sec. 4.1 (fig. 1 right). We used these images to train $\mathcal{V}^2$ and, as for $\mathcal{A}$ (sec. 4.2), we took the CNN pre-trained on the ImageNet classification challenge and replaced its original 1000-way `fc8` classification layer with a 4-way `fc8`. During stage $\mathcal{T}^3$, the viewpoint classifier is useful to select an appropriate location model for object images $\mathcal{O}^{voc}$. Given an input image, we select the viewpoint with the highest probability. Note that the PASCAL VOC 2010 dataset has manual viewpoint annotations for some objects ($\sim 60\%$ for the classes we consider). We use these annotations in sec. 5.2 to evaluate how well our viewpoint classifier $\mathcal{V}^2$ works.

4.5 Mining for new part instances

In stages $\mathcal{T}^2$ and $\mathcal{T}^3$ we mine for new part instances in $\mathcal{O}$ and $\mathcal{O}^{voc}$, respectively. For simplicity, we describe the process to mine from $\mathcal{O}$. Given each set of images $\mathcal{O}_j \in \mathcal{O}$

TABLE 1: Viewpoint classification results (average precision).

	Bicycle	*Bird*	*Bus*	*Car*	*Cat*	*Cow*	*Dog*	*Horse*	*Sheep*	mean
$\mathcal{V}^2$	51.0	48.1	58.4	43.2	59.0	61.1	52.4	53.3	57.3	**53.7**
$\mathcal{V}^{FS}$	42.6	39.3	57.8	38.6	55.4	57.9	51.8	44.5	55.9	49.3

showing viewpoint v_j, we mine positives for part p_i using the appearance model $\mathcal{A}_i^1$ and the location model $\mathcal{L}_{i,j}^2$. For each image, we score all its part proposals with $\mathcal{A}_i^1$ and $\mathcal{L}_{i,j}^2$, perform non-maximum suppression and keep only the proposals with high score. We repeat this for all $\mathcal{O}_j \in \mathcal{O}$ and obtain our final set of new samples. Importantly, we mine for new part instances within object bounding boxes only. Even though the initial appearance models $\mathcal{A}^1$ were trained on 25 samples only, they still manage to localize new part instances, as the search space is very limited. Moreover, this process is able to mine new instances that look significantly different than those in the initial set of easy examples $\mathcal{P}$ (e.g. a frontoparallel wheel against a white background vs a out-of-plane rotated wheel on an actual car, fig. 5), as new part instances can be selected if at the right location according to $\mathcal{L}$, even when $\mathcal{A}$ is not confident about them.

Mining from $\mathcal{O}^{voc}$ is analogous, but requires an extra step, where we use the viewpoint classifier $\mathcal{V}^2$ (sec. 4.4) to predict the otherwise unknown viewpoints of objects $\mathcal{O}^{voc}$.

5 Experiments and Conclusions

5.1 Datasets

We evaluate our framework and all its intermediate stages on the recent PASCAL-Part dataset [1], which augments PASCAL VOC 2010 [26] with pixelwise semantic part annotations. For evaluation we fit a bounding-box to each part segmentation mask. Finally, the dataset contains a `train` and a `validation` subsets. We mine new part instances from `train` in stage $\mathcal{T}^3$, and measure the performance of our framework on `validation`. We verified by using a near-duplicate detector that none of the images we collected from Google Images are in Pascal Parts.

We evaluate on nine diverse object classes (*bicycle, bird, bus, car, cat, cow, dog, horse, sheep*), three parts each (table 2). We treat each leg as a separate instance, rather than grouping them into a 'super-part' (as done by [1]). Note how previous works evaluating on PASCAL-Parts consider fewer classes/parts [1], [5], [35] and operate in a fully supervised scenario (training from manual part location annotations).

5.2 Viewpoint prediction

We evaluate here our viewpoint classifier $\mathcal{V}^2$ trained purely from Google Images (sec. 4.4). We compare it against a viewpoint classifier $\mathcal{V}^{FS}$ trained using manual annotations from PASCAL VOC 2010 `train`. In both cases we use the same CNN model and training procedure (sec. 4.4). We evaluate both classifiers in terms of accuracy on `validation` (table 1). Results show that our viewpoint classifier $\mathcal{V}^2$ considerably outperforms the fully supervised classifier $\mathcal{V}^{FS}$. Results are not surprising, as objects in the PASCAL VOC dataset appear often truncated or occluded and sometimes labelled with the wrong viewpoint. Instead, the images from Google have clean objects with well defined viewpoints (fig. 1). Moreover, objects appear as a whole, leading to better prediction performance.

5.3 Part localization

In this section we evaluate how good our part models are at localizing parts in novel images. We evaluate part localization in terms of average precision (AP) on the PASCAL-Part `validation` set (which was never seen by our learning procedure). As in [1], a part is considered correctly localized if is has an $IoU \geq 0.4$ with a ground-truth bounding-box.

Our location models are conditioned on the object viewpoint, which is however unknown for the objects in `validation`. We apply our viewpoint classifier $\mathcal{V}^2$ on all objects in `validation` to select what location model to use. When detecting parts we use a linear combination of the score given by the appearance and location models ($\mathcal{A} + \mathcal{L}$).

We evaluate each component of our system at each stage of the learning, from $\mathcal{T}^0$ to $\mathcal{T}^3$. For comparison, we trained additional part appearance models $\mathcal{A}^0$ directly on images retrieved by Google, *before* fitting a bounding-box around each training instance (sec. 4.1). Results are presented in table 2 and fig. 5. Naively using images as returned by Google Images ($\mathcal{A}^0$) leads to an AP of only 12.9. This reveals how challenging is the task of localizing object parts on a dataset like PASCAL VOC. Our refined models $\mathcal{A}^1$ perform already better and improve $\mathcal{A}^0$ by +2.8, showing that our polishing process is useful and provides cleaner examples that lead to better performance. The really interesting leap however is achieved in stage $\mathcal{T}^2$ when our framework associates the object to its parts and learns the connection. More precisely, learning the location of the parts under the different viewpoints increases AP to 18.3 ($\mathcal{A}^1 + \mathcal{L}^2$). Using this information to mine for more part instances and update the appearance model improves performance even further to 22.0 ($\mathcal{A}^2$). Ultimately, the combination of these two models ($\mathcal{A}^2 + \mathcal{L}^2$) brings the performance to 23.5. This is almost double the initial AP of naively training detectors directly from part images ($\mathcal{A}^0$). Importantly, at this point we have a complete class model (appearance, location, viewpoint) trained entirely and automatically from Google Images. Finally, if we additionally migrate to the PASCAL VOC domain ($\mathcal{T}^3$) and adapt appearance and location models to it, the performance further improves to a final AP of 27.2 ($\mathcal{A}^3 + \mathcal{L}^3$). The steady improvement exhibited from stage $\mathcal{T}^0$ to $\mathcal{T}^3$ by our incremental learning framework demonstrates its potential in learning complex part models automatically.

For reference, we train two fully supervised models: $\mathcal{A}^{FS1}$ and $\mathcal{A}^{FS2}$. The former uses manual part bounding-boxes from PASCAL-Part `train`, while the latter also uses the part instances from $\mathcal{T}^0$ collected from the web. Similarly to sec. 4.2, for each class we took AlexNet CNN and replaced its last layer with a $(P+1)$-way `fc8` (P parts an one background class). These models provide an upper-bound on what can be achieved by any weakly supervised procedure on this dataset. Our final part detector achieves 27.2 AP, which is 62% of the performance of $\mathcal{A}^{FS1}$. This is very encouraging, given that we train without part location annotations, whereas $\mathcal{A}^{FS1}$ trains from 15K part bounding-boxes on PASCAL-Part `train` (covering our 9 classes with 3 parts each). These take a lot of time as the parts are small and difficult to annotate. Interestingly, $\mathcal{A}^{FS2}$ performs a little worse than $\mathcal{A}^{FS1}$, despite being trained from more data. We attribute this to the difference between the type of images in PASCAL-Part and on the web.

TABLE 2: Part detection results (average precision) on the `validation` set of PASCAL-Part dataset.

		$\mathcal{T}^0$ $\mathcal{A}^0$	$\mathcal{T}^1$ $\mathcal{A}^1$	$\mathcal{A}^1+\mathcal{L}^2$	$\mathcal{T}^2$ $\mathcal{A}^2$	$\mathcal{A}^2+\mathcal{L}^2$	$\mathcal{A}^3$	$\mathcal{T}^3$ $\mathcal{A}^3+\mathcal{L}^2$	$\mathcal{A}^3+\mathcal{L}^3$	$\mathcal{A}^{FS1}$	$\mathcal{A}^{FS2}$	LEVAN [23]	NEIL [22]
Bicycle	*Wheel*	37.2	39.6	50.5	53.9	58.7	56.6	64.0	63.9	75.7	74.2	24.7	43.1
	Saddle	4.2	9.8	14.3	14.0	14.5	17.2	20.6	20.7	35.5	31.7	-	-
	Handlebar	2.2	5.9	3.5	5.9	5.9	8.5	8.7	9.9	25.6	21.1	-	-
Bird	*Head*	16.4	16.4	13.5	22.4	21.0	22.7	22.6	22.7	55.3	52.0	-	-
	Torso	2.6	5.2	10.1	38.0	48.4	48.9	53.7	55.5	60.8	56.3	-	-
	Tail	0.2	0.8	2.0	0.5	0.5	1.4	2.1	2.0	8.7	5.1	-	-
Bus	*Frontside*	40.6	43.1	59.5	60.5	68.2	65.2	69.1	69.3	82.2	80.0	-	-
	Headlight	0.8	1.8	2.1	2.6	2.9	3.5	3.8	4.0	25.5	21.2	-	-
	Wheel	15.1	19.8	18.2	23.1	22.9	27.1	27.0	27.5	50.6	47.3	5.3	4.9
Car	*Backside*	13.8	14.7	20.0	18.6	28.4	23.2	28.5	28.4	43.7	42.2	-	-
	Licence Plate	15.3	15.3	12.6	15.5	15.0	20.5	20.2	21.5	41.0	38.4	5.2	-
	Wheel	20.2	22.2	19.2	26.5	26.5	30.4	30.4	31.0	59.3	59.2	5.5	16.4
Cat	*Head*	25.4	36.9	36.6	48.8	48.2	54.7	54.1	54.6	82.2	80.5	10.9	-
	Eye	10.0	10.0	10.5	16.7	16.7	21.2	21.2	21.4	45.6	43.9	1.4	-
	Tail	0.1	0.4	1.1	1.5	1.6	2.3	2.2	2.4	15.5	9.8	-	-
Cow	*Head*	29.2	31.8	31.8	39.2	39.3	47.1	47.3	47.9	64.9	64.9	35.3	-
	Horn	3.9	6.1	7.2	10.2	10.8	14.5	14.9	15.1	23.1	22.0	-	-
	Muzzle	7.8	9.9	12.1	17.1	17.4	20.4	20.8	21.1	41.6	40.7	-	-
Dog	*Head*	30.1	32.7	32.8	42.3	42.9	56.1	56.5	56.4	79.1	79.9	20.9	-
	Eye	6.5	6.5	7.1	8.1	8.9	10.6	10.6	10.9	17.8	16.4	-	-
	Tail	0.1	0.2	0.8	1.2	1.1	1.4	1.5	1.7	9.4	6.4	-	-
Horse	*Head*	30.7	33.8	33.5	35.7	34.9	37.9	37.2	37.4	64.4	63.9	22.2	-
	Torso	8.1	16.0	46.1	47.2	53.2	48.4	55.7	59.4	71.9	68.9	-	-
	Leg	0.6	12.0	14.7	4.6	5.3	4.6	6.9	7.1	14.3	11.5	-	-
Sheep	*Head*	25.4	28.2	28.6	33.1	33.5	35.2	35.7	35.6	49.2	49.0	18.4	-
	Horn	1.9	2.9	3.3	2.7	3.2	3.1	3.5	3.6	27.3	23.1	-	-
	Leg	0.4	1.1	2.4	3.1	3.5	3.2	3.6	3.8	13.3	11.6	-	-
	mAP	**12.9**	**15.7**	**18.3**	**22.0**	**23.5**	**25.4**	**26.8**	**27.2**	**43.9**	**41.5**	-	-

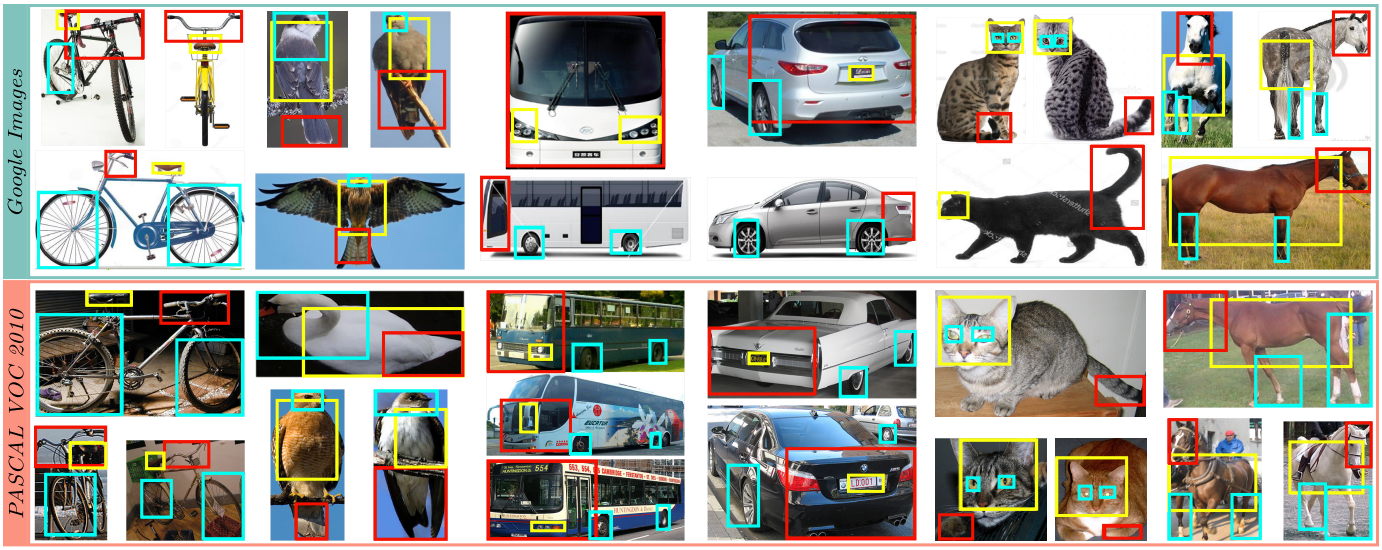


Fig. 5: *Detections obtained by running $\mathcal{A}^1+\mathcal{L}^2$ on object images from Google (top) and $\mathcal{A}^3+\mathcal{L}^3$ on objects from PASCAL-Parts (bottom).*

Comparison to LEVAN [23] and NEIL [22]. LEVAN and NEIL learn detectors from the web and their original papers do not present quantitative evaluation on part detection. Nonetheless, a few of their models represent semantic parts. We evaluate them in this section, using their DPM models [3] they released online [22], [23]. LEVAN learns multi-component object class detectors. The components within each object model are labelled with a name, like 'horse jumping' or 'horse head'. We downloaded the detectors for our nine object classes and selected all components matching our parts. For example, to detect *car licence plate* we run the models labelled as 'plate_car_super3' and 'plate_car_super6'. NEIL, instead, learns a collection of separate models, some representing object classes and others part classes, as well as part-of relation between them. We downloaded all NEIL's models and selected those in a part-of relation with any of the object classes we consider. This only matches one part *wheel*. Only one generic wheel model is available, not associated to a specific object class.

We run all these part models on PASCAL-Part `validation` and show results in table 2 (rightmost two columns). Note how most of the parts we consider are missing from the components learned by NEIL and LEVAN. On the few parts that they learned, our part detectors outperform LEVAN and NEIL by a large margin. The main reason is that their models are trained from part instances downloaded from the web with no (or minimal) refinement: LEVAN uses instances similar to our $\mathcal{T}^0$, and NEIL uses something in between our $\mathcal{T}^1$ and $\mathcal{T}^2$. A second reason is that LEVAN and NEIL's components are based on simple HOG features, which are weaker than CNNs.

5.4 Object detection

In this section we augment the R-CNN object class detector [27] with our part models. The standard R-CNN detector scores each object proposal w in an image with a root filter $\mathcal{R}$ covering the whole object. Inspired by [3] we add a collection of parts arranged in a deformable configuration:

$$\text{score}(w) = \mathcal{R}(w) + \sum_i^N \max_{w' \in \Upsilon} (\alpha_i \cdot \mathcal{A}_i(w') + \beta_i \cdot \mathcal{L}_{i,\mathcal{V}(w)}(w')) \quad (3)$$

TABLE 3: Object detection results (average precision).

Model	Test VOC	*Bicycle*	*Bird*	*Bus*	*Car*	*Cat*	*Cow*	*Dog*	*Horse*	*Sheep*	mean
R-CNN$_1$	2010	64.6	46.8	63.5	56.3	69.0	45.4	62.4	55.1	54.8	57.4
R-CNN$_2$	2010	64.1	43.7	62.5	56.1	67.3	45.1	61.2	55.0	54.2	56.6
R-CNN$_1$ + parts	2010	66.9	49.3	65.6	58.4	70.8	46.8	64.3	58.0	55.9	**59.6**
R-CNN$_1$	2012	63.5	44.4	62.2	55.5	68.1	44.6	61.0	53.5	55.4	56.5
R-CNN$_1$ + parts	2012	66.1	47.2	64.1	58.0	69.6	45.9	62.8	56.7	56.4	**58.5**

where Υ is the set of part proposals inside w. For each part i, the max operation looks for the best fitting proposal $w' \in \Upsilon$ according to the part appearance model ($\mathcal{A}_i$) and location model ($\mathcal{L}_i$), measuring how likely part i is to appear at the location w'. As we have a separate location model per viewpoint, we use our classifier $\mathcal{V}$ to select which one to use on w. We use the same set of proposals for both objects and parts (sec. 4.1). We set the weights α and β by cross-validation on `train`. This overall object class model is similar to [3], but instead of using Gaussian part location models, we have a full probability distributions given by kernel density estimators (eq. 2).

We train the root filer on PASCAL-Parts `train` as in [27]. The other elements of the model are learned from the web and PASCAL-Parts using our technique (sec. 4), i.e. $\mathcal{A}^3$ as part filters, $\mathcal{L}^3$ as location models and $\mathcal{V}^2$ as viewpoint classifiers. No manual part location annotations is used for training. We report object detection results on `validation` of PASCAL VOC 2010 and 2012, in terms of AP (table 3). Compared to using the R-CNN root filter alone (R-CNN$_1$), adding parts increases its performance by 2-3% on all classes, and on both test sets (R-CNN$_1$+ parts). This shows that our part models can help object class detection, even when added to an already strong fully supervised detector like R-CNN. This is an interesting result, especially considering that our parts are designed to be *semantic*, as opposed to discriminative arbitrary patches [2], [3].

For a fully fair comparison, we also train another R-CNN model on object instances from both PASCAL-Part *and* the images we downloaded from the web ($\mathcal{T}^0$). Interestingly, training using this additional data decreases performance by 0.8% (R-CNN$_2$). Again, we attribute this to the difference between the type of images in PASCAL-Part and on the web.

6 Conclusions

We presented a technique for learning part-based models from the web. It operates by collecting object and part instances and by automatically connecting them in an incremental learning procedure. Our models encompass the appearance of parts and their spatial arrangement on the object, specific to each viewpoint. We reported results on the challenging PASCAL-Parts which show that our technique is able to learn good part detectors from the web. Finally, we demonstrated the value of our part models by enriching the R-CNN object detector with parts.

Acknowledgments. Support by ERC Starting Grant VisCul.

References

[1] X. Chen, R. Mottaghi, X. Liu, S. Fidler, R. Urtasun, and A. Yuille, "Detect what you can: Detecting and representing objects using holistic models and body parts," in *CVPR*, 2014.

[2] I. Endres, K. Shih, J. Jiaa, and D. Hoiem, "Learning collections of part models for object recognition," in *CVPR*, 2013.

[3] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," *IEEE Trans. on PAMI*, vol. 32, no. 9, 2010.

[4] P. Arbeláez, B. Hariharan, C. Gu, S. Gupta, L. Bourdev, and J. Malik, "Semantic segmentation using regions and parts," in *CVPR*, 2012.

[5] J. Wang and A. Yuille, "Semantic part segmentation using compositional model combining shape and appearance," in *CVPR*, 2015.

[6] J. Liu, A. Kanazawa, D. Jacobs, and P. Belhumeur, "Dog breed classification using part localization," in *ECCV*, 2012.

[7] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. Jawahar, "Cats and dogs," in *CVPR*, 2012.

[8] N. Zhang, J. Donahue, R. Girshick, and T. Darrell, "Part-based r-cnns for fine-grained category detection," in *ECCV*, 2014.

[9] J. Liu, Y. Li, and P. N. Belhumeur, "Part-pair representation for part localization," in *ECCV*, 2014.

[10] M. Sun and S. Savarese, "Articulated part-based model for joint object detection and pose estimation," in *ICCV*, 2011.

[11] N. Ukita, "Articulated pose estimation with parts connectivity using discriminative local oriented contours," in *CVPR*, 2012.

[12] L. Bourdev and J. Malik, "Poselets: Body part detectors trained using 3d human pose annotations," in *ICCV*, 2009.

[13] N. Zhang, R. Farrell, F. Iandola, and T. Darrell, "Deformable part descriptors for fine-grained recognition and attribute prediction," in *ICCV*, 2013.

[14] G. Gkioxari, R. Girshick, and J. Malik, "Actions and attributes from wholes and parts," in *ICCV*, 2015.

[15] M. Juneja, A. Vedaldi, C. Jawahar, and A. Zisserman, "Blocks that shout: Distinctive parts for scene classification," in *CVPR*, 2013.

[16] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman, "Learning object categories from google's image search," in *ICCV*, 2005.

[17] S. Vijayanarasimhan and K. Grauman, "Keywords to visual categories: Multiple-instance learning for weakly supervised object categorization," in *CVPR*, 2008.

[18] L.-J. Li and L. Fei-Fei, "Optimol: automatic online picture collection via incremental model learning," *IJCV*, vol. 88, no. 2, pp. 147–168, 2010.

[19] F. Schroff, A. Criminisi, and A. Zisserman, "Harvesting image databases from the web," *IEEE Trans. on PAMI*, vol. 33, no. 4, pp. 754–766, 2011.

[20] Q. Li, J. Wu, and Z. Tu, "Harvesting mid-level visual concepts from large-scale internet images," in *CVPR*, 2013.

[21] X. Chen, A. Shrivastava, and A. Gupta, "Neil: Extracting visual knowledge from web data," in *ICCV*, 2013. [Online]. Available: www.neil-kb.com

[22] S. Divvala, A. Farhadi, and C. Guestrin, "Learning everything about anything: Webly-supervised visual concept learning," in *CVPR*, 2014. [Online]. Available: levan.cs.washington.edu

[23] X. Chen and A. Gupta, "Webly supervised learning of convolutional networks," in *CVPR*, 2015.

[24] D. Novotny, D. Larlus, and A. Vedaldi, "Learning the semantic structure of objects from web supervision," in *ECCV workshop on Geometry Meets Deep Learning*, 2016.

[25] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes (VOC) Challenge," *IJCV*, 2010.

[26] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *CVPR*, 2014.

[27] D. Lin, X. Shen, C. Lu, and J. Jia, "Deep lac: Deep localization, alignment and classification for fine-grained recognition," in *CVPR*, 2015.

[28] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut: interactive foreground extraction using iterated graph cuts," *SIGGRAPH*, vol. 23, no. 3, pp. 309–314, 2004.

[29] D. Kuettel and V. Ferrari, "Figure-ground segmentation by transferring window masks," in *CVPR*, 2012.

[30] A. Rosenfeld and D. Weinshall, "Extracting foreground masks towards object recognition," in *ICCV*, 2011.

[31] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *IJCV*, 2013.

[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[33] Y. Jia, "Caffe: An open source convolutional architecture for fast feature embedding," http://caffe.berkeleyvision.org/, 2013.

[34] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Hypercolumns for object segmentation and fine-grained localization," in *CVPR*, 2015.